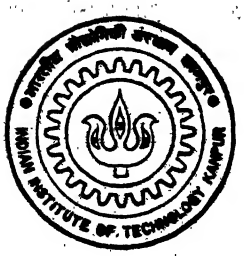
Descriptor Based Information Retrieval

by

Prasad M. Limaye



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Indian Institute of Technology Kanpur

MARCH, 1998

Descriptor Based Information Retrieval

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

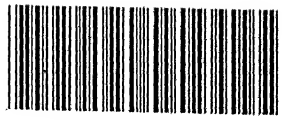
by
Prasad M. Limaye

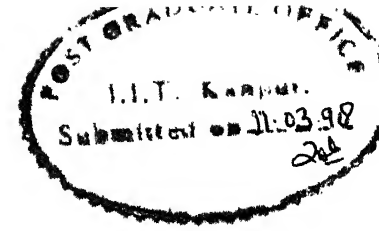
to the
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
March, 1998

CSE-1998-M- LIM- DES

Entered in the System

NiSha
1-6-98





Certificate

Certified that the work contained in the thesis entitled "*Descriptor Based Information Retrieval*", by Mr.*Prasad M. Limaye*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Dr. Harish Karnick)
Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.

March, 1998

Abstract

The exponential growth of information and its availability in digital form has made the task of retrieving relevant information significantly more difficult. Navigating such large information spaces to retrieve relevant information has three major hurdles :

1. Lack of good and appropriate descriptors for a certain piece of information.
2. Lack of mechanisms/tools to express user needs or interest profile precisely and
3. Lack of good quality interfaces which gather user feedback and appropriately narrow or broaden the interest profile to make it a better filter for the data.

Researchers have investigated many different techniques to increase the relevance of the results of information retrieval systems. One approaches has been automatic generation of a Thesaurus or Network of Concepts from databases.

This thesis discusses a new model for a typical bibliographic database and an incremental clustering algorithm based on the model. The algorithm produces a hierarchy of conceptual clusters which could be used to aid an information seeker to define and refine his interest profile.

The algorithm was implemented and it's behavior observed. It performs reasonably well at the lower levels of the hierarchy.

Acknowledgment

I am highly indebted to my thesis supervisor Dr. H. Karnick for his constant encouragement and guidance throughout my work. Inspite of his various involvements, he always had time for me.

My sincere thanks to the Library Automation Staff, especially Mr. Rawat and Mrs. Sengupta for their help and suggestions during the experimentation phase. I also wish to thank CSE lab staff for their assistance during the thesis work.

It is really difficult to put into words my gratefulness towards my parents who are a constant source of inspiration and support for me.

I would like to thank all my classmates, who directly or indirectly helped me complete the thesis work. I would also like to thank all my Hall-4 friends, especially Rajendra, Ganesh, Girish, Amol and Adish who made me feel at home during my stay here in IIT Kanpur.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 The Basic Problem 2

1.3 Typical Intelligent IRS 3

1.4 The Aim 4

1.5 Thesis Organization 5

2 Related Work **6**

2.1 Internet Search Engines 6

2.2 Processing of the Search Engine Results 7

2.3 Improvements in the User Specifications 8

2.4 Use of Thesauri 11

2.5 Automatic Construction of a Thesaurus 13

3 The Model **17**

3.1 Information Delivery Vs Information Retrieval 17

3.2 The Domain 18

3.3 The Proposed Model 19

4 The Algorithm **24**

4.1 Basic Terminology 24

4.2 Greedy Approach 25

4.3 Repeated Splitting Approach 26

4.4 Incremental Approach 28

4.5 The Proposed Algorithm 30

4.6 A Case Study 32

5 Implementation and Results **35**

5.1 Database Preprocessing 35

5.2 Implementation Overview 36

5.3 Database Connectivity - *JDBC* 38

5.4 Results 40

5.4.1 Process Trends 40

5.4.2 Cluster Hierarchy 45

6 Conclusion and Future Work **50**

6.1 Summary 50

6.2 Conclusive Remarks 50

List of Tables

1 Distribution of descriptors in the cluster hierarchy 46
2 Distribution of descriptors in the cluster hierarchy - *Cont'd.* 47

List of Figures

1 A block diagram of typical Intelligent Information Retrieval System . 4
2 Search tactics suggested by Bates 8
3 Natural Language Interface used by EP-X 10
4 Automatic Thesaurus Generation using Term Discrimination Model by Crouch et al. 15
5 Frame based thesaurus by Chen and Lynch 16
6 Structure of Document - Descriptor Domain 18
7 New model for Document - Descriptor Domain 21
8 Comparison of the pairwise similarity criteria and the new similarity criteria 22
9 An Overview of Learning Algorithm 28
10 Possible rearrangement for stabilizing cluster hierarchy 30
11 Part of Hierarchical Knowledge affected by a new document 31
12 Trend - Number of descriptors in the hierarchy 40
13 Trend - Number of m-b-nodes 41
14 Trend - Number of checks for rearrangement in the hierarchy 42
15 Trend - Number of rearrangements done in the hierarchy 43
16 Trend - Overall (total) number of rearrangements done in the hierarchy 43
17 Trend - Processing Time 44
18 Trend - Number of clusters in the Hierarchy 44
19 Resultant cluster hierarchy 49

Chapter 1

Introduction

1.1 Motivation

In the past few decades we have seen rapid growth of relational, textual and graphical databases. The availability of cheap and effective storage devices have led to easy information collection and storage. Optical character recognition, word processors and computer publishing software are capable of producing massive quantities of online text. The Internet has made the problem more acute because new information is produced or existing information is modified in time-spans which could sometimes be of the order of minutes. In addition, the information is usually distributed at many geographical locations making it extremely difficult to have proper classification and categorization. As a result, the amount of effort required to retrieve relevant information, which is proportional to the amount of information stored, has also increased considerably. Due to lack of explicit semantic clustering, classification and categorization, conventional keyword driven search techniques have become inadequate. It has been observed that only the users with extensive subject area knowledge, system knowledge and classification scheme knowledge are able to explore in these databases.

Researchers in expert systems and artificial intelligence have opened up a new area called *Knowledge Mining* or *Knowledge Discovery in Databases*[KDD] to cope with these problems. Among the significant facts about the large scale real world databases is the amount of domain knowledge hidden in voluminous data. KDD

combines techniques from machine learning, pattern recognition and statistics to automatically extract concepts and concept interrelations from the voluminous data. This extracted knowledge is subsequently used to support human decision-making. Many recent results have been reported on extracting different kinds of knowledge from databases including drug side effects, fraud detection, diagnostic rules and others.

1.2 The Basic Problem

The two main problems faced by a user while searching for relevant material in databases are -

- A small fraction of relevant information available in the database is retrieved.
- The fraction of the retrieved information which is relevant is low.

Most attempts to build a better IRS (Information Retrieval System) try to improve these two ratios. A term **recall** has been coined for the proportion of relevant material retrieved and **precision** for the proportion of retrieved material that is relevant [BM85].

One major reason for poor recall and precision is the inability of a user to specify his needs. A typical user often needs help in defining or refining his topics of interest[PJSC89]. Some need to learn more about the topic area in order to decide what is interesting and what is not. Others have difficulty in expressing their interest (even in natural language). Still others have a clearly defined and stated topic but find that it is far too broad (or too narrow) retrieving too many (or too few) results to be acceptable. [PJSC89] describes searching as a combination of a learning task and a prediction task. First, the user must predict how semantic content of relevant documents will be described in the document and then predict the specific language that will be used to represent the intended meaning fully and correctly. Finally, as the search progresses, the user's notion of what he requires becomes clearer to him and may shift in emphasis.

Borgman[Bor86] identifies two types of knowledge necessary to search -

- The knowledge of mechanical aspects of searching (e.g syntax and semantics of both the query language and the system interaction commands)
- The knowledge of the conceptual aspects (e.g ways to broaden and narrow searches using alternative vocabulary, choosing alternative search paths).

One other aspect related to poor recall and precision, is poor assignment of descriptors to documents. Descriptors are supposed to represent a document (semantic contents of the document). The search engine matches the user profile with the descriptors of documents and not the actual documents (matching with the documents is impossible, anyway because we do not even know exactly what it means). Incorrect selection of descriptors for documents may thus lead to its incorrect classification and this may deteriorate the recall (the data will not be retrieved if needed) and precision (the data will be retrieved even if not needed) because of the match between user queries and the descriptors which do not relate to the semantic content of the data.

1.3 Typical Intelligent IRS

The following figure shows a block diagram of a typical IRS employing a knowledge base to improve the quality of search results.

At the front end is a user interface which interacts with the user to enable him define and refine a representation of what the user indicates as his interest. This representation is called the user profile. The profile is used by the search engine to search the database and fetch the results. Some post-processing is then done on the results and the results are presented to the user. The results can be used by the user to refine or review his profile. Additionally, a knowledge base is generally used to help user get an insight into the data-space. This knowledge base acts as a source of the domain knowledge (the domain of the data in the database) which is often needed by the user to understand his profile more clearly.

In the simplest form of an IRS, the user interface is a command interpreter to which the user specifies queries in some standard form. The query is stored as the

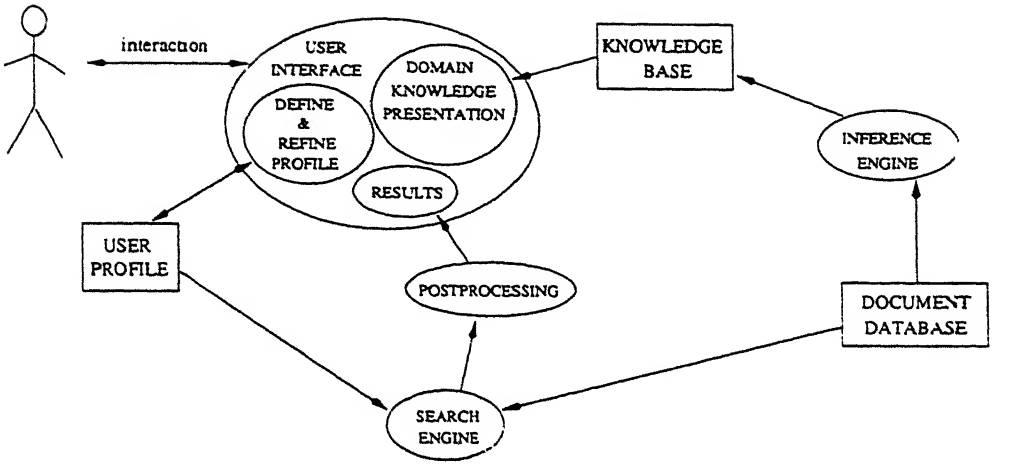


Figure 1: A block diagram of typical Intelligent Information Retrieval System

user profile and directly given as input to the search engine. Post-processing is absent and the results are directly notified to the user.

1.4 The Aim

A searching process is governed mostly by three major issues. Any system used to enhance relevance of information retrieved, can be categorized into one of the following categories each of which addresses one of the issues.

1. **Organizing the document database.** The issues which are addressed here include - What descriptors are appropriate (descriptors which represent the semantic contents of documents.) for each of the documents in the databases? Can there be a system for automatically assigning appropriate descriptors to documents, etc.

   Major research in this category is based on use of natural language processing techniques to automate the process of descriptor assignment.

2. **Modeling the user.** Considering a user's knowledge of the area which is being searched, and his ability to define the search conditions and subsequently narrow or broaden the search condition to get better results in terms of recall

and precision, can we model a user searching the database with what we call a user's interest profile? What form of profile will it be? etc.

Any utility which attempts to model a user must be able to do two things :

(a) Give an effective way for a user to state his interest profile.

(b) Use the interest profile to create a filter which allows relevant information to reach the user.

Modeling a user with an interest profile is particularly hard to achieve since a user wants only that information which is relevant but at the same time does not want to miss any information which is likely to be relevant. So in a sense the filter must be simultaneously broad and narrow.

3. **Use of Supplementary Tools and Techniques.** The problems in this category include - What search tools like search engines and subsidiary knowledge bases like catalogs or thesauri could be used to help a user and in what way they should be used.

Since keyword based document description or indexing is widely used, a system which makes use of the keyword descriptors to extract knowledge about the concepts hidden in a database seems possible. This thesis seeks to come up with one such keyword based concept extraction technique. The technique could be used to build a knowledge base which will facilitate easy modeling of user interest profile. In a sense, the attempt falls in the third category above but the second category viz. 'Modeling the user' is also considered.

1.5 Thesis Organization

In Chapter 2 various techniques used to improve individual components of IRS have been discussed. Chapter 3 discusses in detail the issues in Automatic Generation of Thesaurus which is the most widely used type of Knowledge Base for a bibliographic databases. Chapter 4 discusses The proposed incremental clustering algorithm. In Chapter 5 we present the results of an implementation of this algorithm. conclusions and future research directions are pointed out in Chapter 6.

Chapter 2

Related Work

Research to improve access to online information is proceeding in many directions. In this chapter we review the major tools and techniques.

2.1 Internet Search Engines

Due to rapid growth of the Internet in terms of the people it reaches and the material it accommodates, search engines for the Internet have become a major issue for The Internet users. Internet search engines fall into the category 'Use of Supplementary Tools and Techniques' discussed in Chapter 1. Some techniques which also model the user are described in next section.

A search engine is a database and tools to generate that database and search it. A related concept is catalog which is an organizational method and related database plus tools for generating it. A search engine finds information for its database by accepting listings sent in by authors wanting exposure or by getting the information from their "Web crawlers", "spiders" or "robot" programs that roam The Internet storing links to and information about each document they visit. Among the notable search engines are AltaVista [http://www.altavista.digital.com], Lycos [http://www.lycos.com], InfoSeek [http://www.infoseek.com].

Almost every search engine allows user to specify boolean queries through the use of words AND, NOT and OR (or similar technique). Search engines give, each document they find, some measure of the quality of the match to the user's search

query - a relevance score. Relevance scores are generally based on the number of times a search term appears in a document, its presence in the title or beginning of the document, the textual distance between search terms in the document, etc. Some engines allow the user to control the relevance score by giving different weights to each search word [Ass].

2.2 Processing of the Search Engine Results

Among the ongoing attempts in Internet search technology are those employing post-processing. Post-processing refers to the options and steps users are offered after they receive initial result lists. No such hit list is ever likely to be just right. In large measure it's accuracy is dependent on how well the user knows both - the topic being searched and the available data. Post-processing can help user specify broader, narrower or just plain different search string as part of a set of ongoing steps rather than as isolated shots in the dark.

Most of the post-processing is interface related. The query by example technique is used in many search engines like Excite [http://www.excite.com] using "More like this" button. Some common things such feedback does is grouping by similar URL, similar phrasing, etc. Post-processing thus requires a whole second search on the data.

Some other forms of post-processing relies on presence of other terms and number of occurrences of these other terms in the vicinity of the search terms in the documents. e.g. excite [http://www.excite.com] processes the results of user queries and fetches out the most commonly found terms in the vicinity of search terms. These terms are then displayed along with the actual results. These terms serve as clues to the user to modify his search criteria.

Meta-searching is also a commonly employed post-processing technique. In such search engines (e.g. MetaCrawler [http://www.metacrawler.com]), a user query is fed to several other search engines and the results are collected. The results are then analyzed to find common hits and relevance is calculated to rank the documents.

There is a new trend towards a visual presentation of search results to help users more easily grasp grouping of data. This visual representation of results is based

on clustering or grouping of the search results. A new front-end called SemioMap [http://www.semio.com] offers some graphical results using a Java-based client that can be hosted on a web-page. For example, if you search for the term "Washington", the search engine would seek to qualify search results into as many identifiable groups as possible "State of Washington", "Washington, D.C.", "George Washington", "George Washington Carver", "Washington Redskins" and so on.

2.3 Improvements in the User Specifications

Another direction of research in terms of Modeling the user is the use of tools and techniques which facilitate improvement in the user specification.

Many discussions on search expertise focus on issues pertaining to use of logical operators to combine search terms. One of the early works in this area is due to Oldroyd & Schroder [OS82]. They discuss the effect of various operators on recall and precision. Bates [Bat79] suggests the use of "idea tactics" to improve searching. She outlines 29 search tactics in four areas : monitoring, file structure, search formulation and term manipulation. The search formulation and term manipulation tactics describe the available techniques to broaden or narrow down queries. However, she includes no guidelines as to when each tactic is appropriate.

```
BRAINSTORM :  Generate many ideas & suspend critical reaction until
              the ideas are well-formed & can be fully evaluated.

CONSULT    :  Ask a colleague for suggestions or information in dealing
              with a search.
```

Figure 2: Search tactics suggested by Bates

It would even be possible to build systems that simply suggest such domain independent tactics or strategies to users when they need help [PJSC89]. Instead most computerized information retrieval systems have been given substantial knowledge about specific domains. This domain knowledge is then used by the system in assisting the searcher to identify relevant documents.

Smith et al. [PJSK84] identify a set of search tactics, including 19 domain dependent tactics. They noted when each of these tactics was applied and whether an expert used the tactic spontaneously or in response to some cue in the retrieved documents. The results of this study are used as the basis for EP-X (Environmental Pollution eXpert), an online search intermediary in searching environmental literature of chemical abstracts. EP-X represents the meanings of concepts and topics in the domain of interesting the form of hierarchy of semantic terms and frames. This knowledge is used to identify and resolve ambiguities in user queries and also to broaden or narrow down the query, using combinations of specific cases of a concept with the concept itself. EP-X uses a natural language dialog with the user to find out possible meanings of user query and to clarify any ambiguities. It also suggests query modifications to the user in order to improve relevance of results, broadening and narrowing the query.

EP-X serves to illustrate several important concepts about thesaurus systems.

- The same set of concepts can be organized into radically different hierarchies based on alternative organizing perspectives like techniques used for some process, base subject on which a technique is based, etc.
- Users are often interested in the relationship among concepts rather than a single concept.

In another attempt, Williams [GS91] has developed an interesting categorization of searching situations with all possible responses, which could be used as an expert system's knowledge base. Based on desired and achieved values of number of documents, recall and precision, 64 search situations have been identified which result in 27 unique states. He has defined 4 variables - generality, exhaustivity, simplicity and ambiguity which can be manipulated in each of the states to achieve desired search results. However, he does not indicate how these manipulation techniques should be combined effectively. Also several states have conflicting manipulations which are difficult to resolve.

Tong and Shapiro describe a rule based system RUBRIC that can represent partial relevance [MG85]. In RUBRIC, production rules are used to represent rule

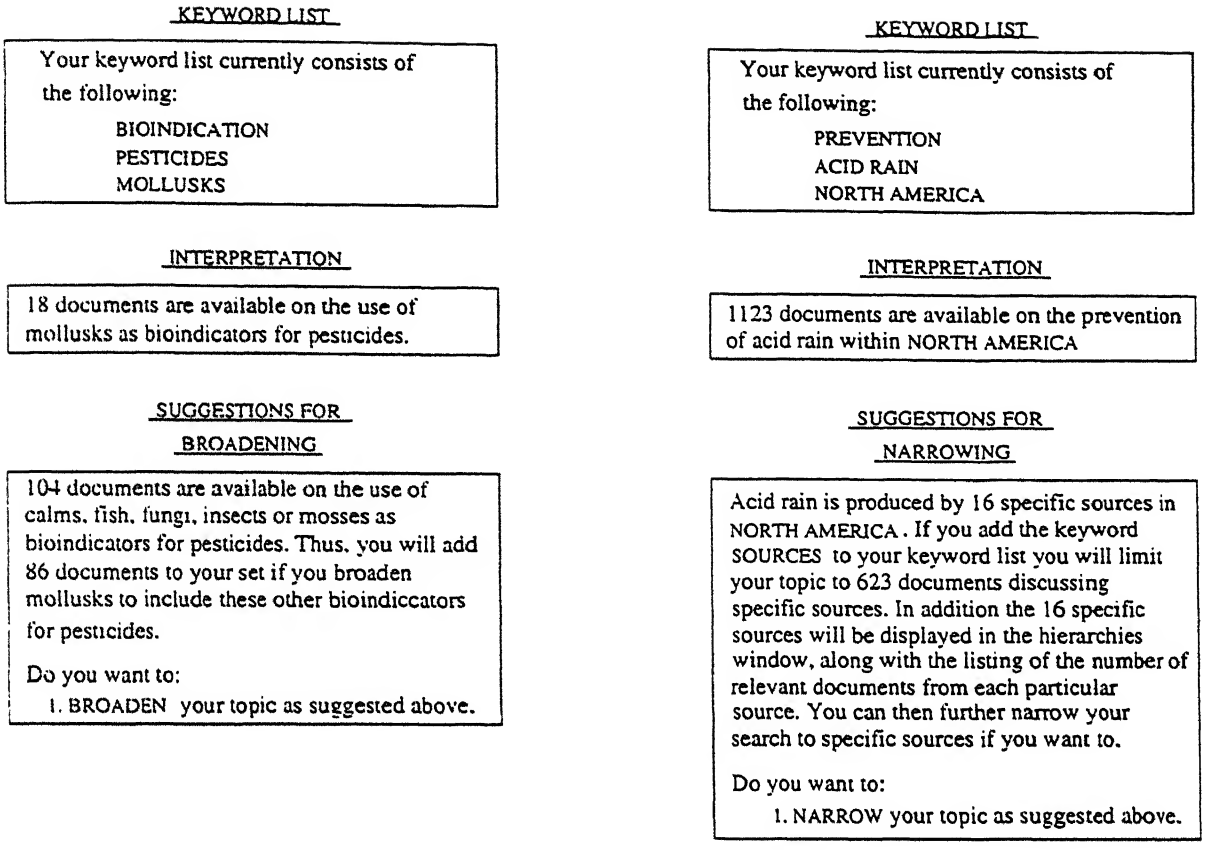


Figure 3: Natural Language Interface used by EP-X

evaluation trees. To determine the relevance of a document to a topic, RUBRIC matches document terms to the leaf nodes of the tree of topics. It then uses associated weightings to compute weight of the topic node which is a measure of the relevance of the document of the topic.

Another study of automatic query reformulation is [GS91] by Gauch & Smith. They categorize query reformulation techniques as :

- Expanding concepts - It involves adding search terms to positive (or negative) concepts to broaden (or narrowing down) the search. Concepts may be expanded by adding synonyms, related search terms and by stemming.
- Adjusting context - Their expert system manipulates four different contexts:

the distance between words in positive and negative multi-word phrases as well as the distance between positive and negative search concepts. Broadening is done by increasing positive contexts and narrowing by increasing negative contexts.

- Changing the query structure - A query can be broadened by two ways
  (a) Positive AND operators can be switched to OR operators (and negative OR operators switched to ANDs).
  (b) The negative part can be dropped altogether.

Gauch & Smith also give an elaborate way to apply these reformulations according to the quality of the results.

2.4 Use of Thesauri

The most commonly used supplementary tool to improve user specification is Thesaurus. An intelligent user interface must have knowledge. Rissland [Ris84] classifies this knowledge into different forms -

- Knowledge of the user's expertise, style and history.
- Knowledge of the available tools being used so that the user does not need to be concerned with interface issues.
- Knowledge of the conceptual hierarchies in the document space.

A thesaurus is one tool satisfying the third requirement above. In this case, the thesaurus also partially relates to the user because the expertise of the user should be partially captured by the information in a good thesaurus. Hierarchical knowledge has always been a base for decision making. A thesaurus exists primarily to link semantically related terms (not just synonyms) and to provide sufficient hierarchical structure.

A typical example of an algorithm using a thesaurus to calculate conceptual closeness between documents and user queries is by McMath et al.[CFMR89]. The thesaurus they used is the computing reviews classification structure (CRCS) of

the association of computing machinery (ACM). In this experiment they tried a graphical presentation of the thesaurus in a windowing environment. Users could browse through the thesaurus displayed as a two-dimensional network and form their queries using thesaurus terms. The results and their relevance were also graphically depicted.

In their experiment, six different formulae were used for calculating semantic distance. All of them calculated minimum path length between query and document terms. The user had an opportunity to choose from these six methods and setting an upper cutoff for the path length. Both the query Q and the document Doc were represented by set of thesaurus terms (t_i, t_j, etc.). The simplest formula for DISTANCE (among the six mentioned) was

$$DISTANCE_{closetdoc}(Q, Doc) = \frac{1}{n} \sum_{t_i \epsilon Q, t_j \epsilon Doc} min\{d(t_i, t_j)\}$$

Weights were assigned to links in the CRCS in each of two directions separately to reflect the human perception more closely. Distances were always measured from the query term to the document term. As a test procedure four queries and nine documents were given to each of fifteen students, each of whom scored each document to each query on a 0 to 99 scale. It was found that each of the six formulae agreed with the students on each of the four queries.

Coalsort [MC87], a knowledge based interface that facilitates the use of bibliographic databases in coal technology, uses a semantic network representing an expert's domain knowledge. The expert system developed by Shoval [Sho85] suggests search terms. The knowledge base is represented as a semantic network in which nodes are words, concepts or phrases. The Intelligent Intermediary for Information Retrieval (I^3R) [CT87] used a blackboard architecture to facilitate communication between a group of "experts". It included a user model builder, a query model builder, a thesaurus expert, a browser expert and an explainer.

Chen, et. al[HCN93] used a blackboard based design and a neural net spreading activation algorithm to span multiple thesauri. Mentioning the philosophy behind the use of the blackboard they state -

> During a search, the user's own subject area knowledge, intermediate search results, thesaurus terms and other incidental cues displayed on

the screen may trigger appropriate search terms and help the user find related documents.

User generated or selected search terms, representing the concepts in the queries, are posted on the "concept blackboard" and documents derived during the search are recorded on the "document blackboard". A user can browse through various thesauri, change his needs, give feedback to retrieved results and act on system suggestions and clues. As a user's queries become more focussed and well articulated after extensive iterations, documents posted on the blackboard have high recall and precision.

2.5 Automatic Construction of a Thesaurus

Experiments have investigated the problem of construction of thesauri by manual, semiautomatic and fully automatic means. Virtually all experiments to automatically construct a dictionary or a thesaurus are based on the **Vector Space Model** [GSY75a]. In this model each document is viewed as a collection (or vector) of words or word types. Document space is thus a vector space of dimension m, where m is the number of word types in the collection. A document vector d_j is represented by a set of terms, d_{jk}, $1 \leq k \leq m$, where d_{jk} represents the frequency or weight of the term k in document j. Similarly, queries are also represented by weighted term vectors. The similarity between two vectors is computed by means of a similarity measure, such as cosine. Given any two term vectors the similarity between them may be assumed to be inversely related to angle between them. The (normalized) vectors become points in a vector space and distance between two such points is inversely related to the similarity of the corresponding vectors. The vector space model facilitates result ranking, relevance feedback and query construction but the major difficulty with it is that the terms are assumed to be independent of one another and term relationships can not be expressed within the model.

Salton et.al[GSY75b] have shown that the best document space for retrieval purposes is the one which maximizes the average separation between documents in document space. This result is the conclusion of their **Term Discrimination**

Model. In this model, which is an extension to the **Vector Space Model**, each term in the vector is weighted according to its discrimination value. The discrimination value of a term is defined as a measure of the change in space separation that occurs when a given term is assigned to the document collection. A good discriminator is a term, that, when assigned to a document, decreases the space density (documents move closer).

Two different approaches have been used to calculate discrimination value - **The exact method**, which involves calculations of all pairwise similarities between the document vectors and **The approximate method**, in which an artificial average document called centroid is constructed and the sum of similarities of each document with the centroid is calculated. Discrimination values produced by these two approaches differ significantly but the ranking of terms is highly compatible.

The exact method is $O(mn^2)$ time complex and the centroid method is $O(mn)$, for a collection of n documents and m word types. Yet for a document space of reasonable size even the centroid approach is expensive. As a reasonable alternative Salton et.al have suggested use of document frequency as an approximation to discrimination value.

Crouch [Cro90] has given details of his experimental research carried out to investigate the feasibility of constructing a thesaurus based on term discrimination value model using a greedy clustering algorithm.

Initially documents to be clustered are viewed as singleton clusters. At each step, the similarity between all clusters is computed. The two most similar clusters are merged to form a new cluster consisting of all documents in both original clusters. The process continues until everything is merged in a single cluster and a hierarchy (a binary tree) is formed.

According to the way in which similarity between two clusters or documents is defined, three types of algorithms have been identified : **Single Link Method** in which similarity between two clusters is defined as the maximum of the similarities among all inter-cluster pairs of documents, **Complete Link Method**, in which the similarity between two clusters is the minimum similarity among inter-cluster pair of documents and **Group Average Link Method** in which similarity is defined as the mean of similarities among all inter-cluster pairwise similarities.

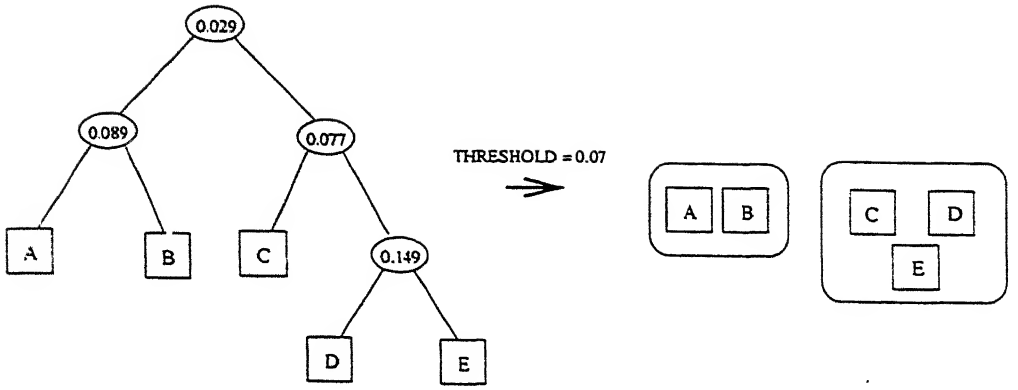


Figure 4: Automatic Thesaurus Generation using Term Discrimination Model by Crouch et al.

The resultant hierarchy is then traversed and thesaurus classes are generated using user supplied parameters such as threshold value of similarity (all 'cluster merges' above this value are merged into a single cluster), number of documents in cluster, minimum document frequency, etc.

Chen & Lynch [CL92] have used what they call the *Cosine Algorithm* to calculate the similarity weight between two descriptors thereby creating a frame based thesaurus. In their *Cosine Algorithm*, they define the similarity weight between two descriptors T_j and T_k as

$$w(T_j, T_k) = \frac{\sum_{i=1}^{n}(d_{ij} X d_{ik})}{\sqrt{\sum_{i=1}^{n} d_{ij}^2 X \sum_{i=1}^{n} d_{ik}^2}}$$

Where n indicates total number of documents in database and d_{ij} indicates descriptor T_j in document i (value 0 or 1). In the same experiment they tried with asymmetric similarity weight criteria. In this approach called *Cluster Algorithm* similarity weight from descriptor T_k is given as

$$w(T_j, T_k) = \frac{\sum_{i=1}^{n}(d_{ij} X d_{ik})}{\sum_{i=1}^{n} d_{ij}}$$

Fig 5 shows the frame-based representation of a keyword produced by the Cosine Algorithm and the Cluster Algorithm. Each frame contains properties of a descriptor (type of descriptor, name and number of occurrences in the database) and its similarity weights with other descriptors, calculated by the formulae just discussed.

```
(object: KEYWORD
    name:                [technology transfer]
    #_of_occurances:     [1742]
    keyword:             [(export controls 0.439) (trade 0.301) (covert 0.222)
                         (export 0.173) (import 0.157) (micro-electronics 0.132)
                         (software 0.113) (microcomputer 0.081) (microprocessor
    folder:              [(ewttknow.dat 0.534)] East-west technology transfer
    person:              nil
    organization:        [(IBM 0.070)]
    country:             [(US 0.265) (Hungary 0.156) (USSR 0.155) (UK 0.135)
                         (Japan 0.119) (Poland 0.092) (Czechoslovakia 0.084)]
)
```

Figure 5: Frame based thesaurus by Chen and Lynch

While discussing the problems of automatic global thesaurus construction Crouch [Cro90] concludes that the *Term Discrimination Value Model* tells us that low frequency terms are better candidates to be grouped in thesaurus classes but the term occurrence is especially difficult to ascertain for low frequency terms where automatic thesaurus construction is of most use.

Attar & Fraenkal investigated automatic construction of what they called *Local Thesaurus* [AF81]. In the approach the information retrieval system constructs a feedback query by adding terms lexically related to search terms. The method is inexpensive & produces searchonyms, terms viewed as synonymous in the context of the particular query.

Chapter 3

The Model

After reviewing earlier approaches, we propose a tool which addresses the problem of automatically generating navigational aids to navigate in the document space. The aid is akin to a thesaurus insofar as it tries to cluster similar descriptors together.

3.1 Information Delivery Vs Information Retrieval

First we focus our attention on the problem of effectively capturing the scope of interest of a user. An effective search can be viewed as a selection process. Consider a scenario in which a user takes a look at each document in the database and marks it relevant or irrrelevent. This procedure is surely impractical because of the time it consumes. It would be a good idea, however, if a user could mark several documents at a time. This is, in fact, the basic principle of clustering. So what is it that is used by a user to categorize a document as relevant or irrelevant; it is not the document as a whole but a combination of the topics it contains. Thus we require the information about what topics are being covered in a given document base, in order to be able to describe the user's interest in a more concise way. And for this we will use clustering.

We want to find out clusters of descriptors which would be coherent in terms of feedback (relevant or irrelevant). More clearly stated, the feedback of a user to a cluster of descriptors should more or less match with his feedback to each of the individual descriptors, irrespective of what type of feedback (relevant or irrelevant or

fuzzy type) it is. This is possible only because both interesting and non-interesting information occurs in groups. In fact it is the particular user's view which makes some clusters interesting (relevant) or non-interesting (irrelevant). So we have a possibility of constantly maintaining such coherent clusters of descriptors and user's feedback to these clusters (which will be the user profile). The information retrieval system thus becomes an information delivery system. As the new information is added to the database it is added to some clusters. According to the user's profile the information can then be delivered to him.

3.2 The Domain

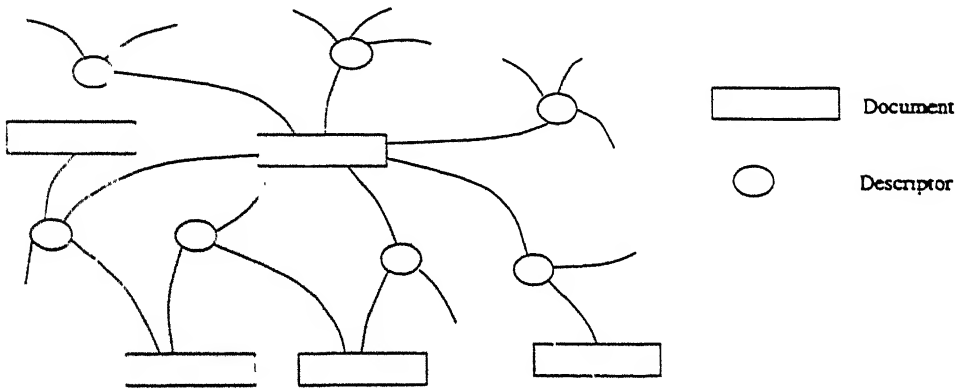


Figure 6: Structure of Document - Descriptor Domain

Before going for a new clustering algorithm, let us investigate the structure of the selected domain and list our assumptions. We select bibliographic databases in a particular field for our experiment. The assumption is that the database will have some good categorization information for the particular field. A general bibliographic database has all the information about published documents - title, authors, abstracts, date of publication, journal name and keywords (sometimes called identifiers, key-phrases or key-terms). All these are in a sense 'descriptors' of documents but we focus our attention only on keyword descriptors. We refer to keyword descriptors as 'descriptors' henceforth to keep in mind that other types of descriptors (i.e authors, abstract etc.) could also be used in the process.

A document has several descriptors. A descriptor may be contained in several documents. Thus, in a bibliographic database, documents and descriptors have many to many relationship and exhibit a network structure as shown in Fig 6.

3.3 The Proposed Model

The *Vector Space Model*, which is a standard model for clustering in document space, considers each document at the same level of generality. Accordingly, the clustering algorithms based on the model form a set of clusters containing related documents and no relationship among the clusters. On the other hand, a typical thesaurus, which is a classification structure in descriptor space, is a hierarchical structure in which related descriptors are combined to form a topic, related topics are combined to form more general topics, and so on. The relationships among the descriptors in such a thesaurus are of three kinds - 'broader than' and 'narrower than' for descriptors in two topics (one contained in other) and 'related to' for the descriptors in a single topic. Hence for clustering in descriptor space, we need to have a hierarchical model of the domain.

To develop a model for generating a hierarchical clusters automatically, we need to define some terms

- **A cluster** is a set of descriptors and smaller clusters which are closely related in document-descriptor space (they share higher number of documents than an average pair of descriptors do).
- **A direct sub-cluster** or just **sub-cluster** of a cluster is a smaller cluster contained in that cluster.
- **An indirect sub-cluster** of a cluster is a smaller cluster contained in a direct sub-cluster or an indirect sub-cluster of that cluster.
- **A fundamental cluster** is a cluster which contains only descriptors and no smaller clusters.
- **A core descriptor** is a descriptor in a fundamental cluster.

- **A bridge descriptor** is a descriptor in a non-fundamental cluster.
- **Unrelated clusters** are a set of clusters such that none of the element clusters is a direct or indirect sub-cluster of any other element cluster.
- **A co-occurrence** is the relation of two or more descriptors due to a document which contains those descriptors.
- **An intra-cluster co-occurrence** is the co-occurrence between descriptors of same cluster.
- **An inter-cluster co-occurrence** is the co-occurrence between descriptors of unrelated clusters
- **A generic-specific co-occurrence** is the co-occurrence between a descriptor of a cluster and a descriptor of a direct or indirect sub-cluster of that cluster, respectively.
- **A document is said to contain a cluster** if it contains at least one descriptor contained in the cluster or any of it's direct or indirect sub-clusters.

In view of the above definitions, a document may contain descriptors from several unrelated clusters. This agrees with our intuitive view - A document typically refers to more than one topic in a thesaurus. After incorporating the concept of clusters, the document-descriptor domain can be visualized as in Fig 7. The clusters and descriptors alone, form a hierarchical structure. At the root is the cluster representing the whole domain. The structure of the domain space at each level of the hierarchy is a network of clusters as in Fig 6, except for the fact that descriptors are replaced by descriptors and clusters. Actually, we can regard a descriptor as a special kind of cluster.

A simple descriptor-descriptor link which earlier represented just co-occurrence can now be classified as intra-cluster co-occurrence, inter-cluster co-occurrence and generic-specific co-occurrence. Given a domain, a very large number of configurations similar to one depicted in Fig 7 are possible. But for a configuration to be of any real use, the number of inter-cluster co-occurrences should be minimum and intra-cluster co-occurrences should be maximized.

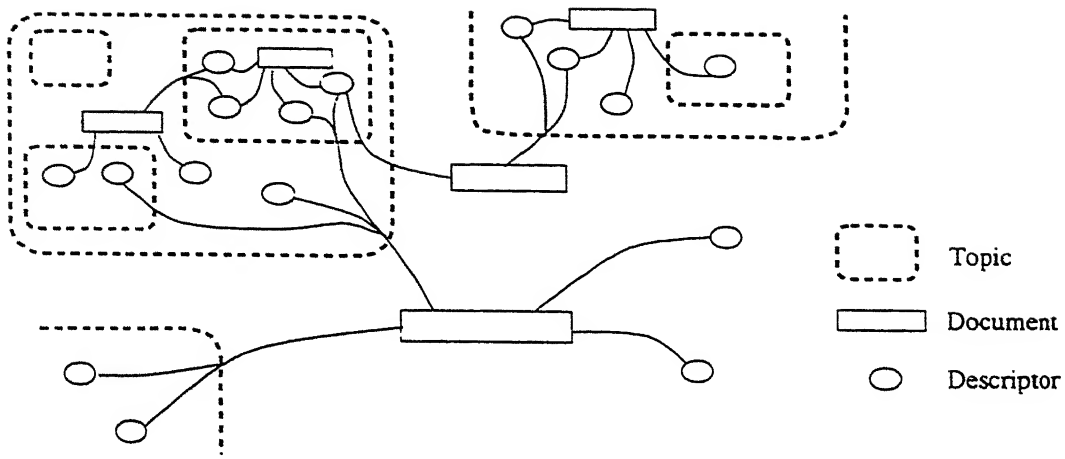


Figure 7: New model for Document - Descriptor Domain

A single document gives rise to a co-occurrence involving one or more (typically more than two) descriptors. In the approach taken by Chen and Lynch [CL92] (refer Chapter 2), only the co-occurrences involving two descriptors are used to calculate similarity weight between those two descriptors. The approach is slightly conservative in the sense that it does not give any information about the co-occurrence between three, four and higher number of descriptors. Hence, it is inappropriate to use the same form of similarity weight when we are trying to cluster possibly more than two descriptors and/or smaller clusters.

A co-occurrence, as one can see from Fig 6, is an n-ary relation and each element of the relation is represented by a document i.e. Each document gives rise to co-occurrence between its n descriptors. To model this correctly, we need a similarity criteria which should be a function of multiple (one, two or more) descriptors or clusters.

We define **similarity weight of a set of descriptors and clusters** as the number of documents containing multiple descriptors and/or clusters from the set. Formally stated,

$$w(S) = \sum_{\forall Doc_i \epsilon D}[(\sum_{\forall Desc_j \epsilon S} d_{ij}) + (\sum_{\forall Clust_k \epsilon S} d_{ik}) > 1]$$

where, D is a set of all documents in the database, d_{ij} indicates presence of

the descriptor $Desc_j$ from set S in document Doc_i (value 0 or 1) and d_{ik} indicates presence of the cluster $Clust_k$ from set S in document Doc_i (value 0 or 1).

Similarly, we define **Compactness of a cluster** as the similarity weight for the set of descriptors and direct-sub-clusters of the cluster.

Consider a document having n descriptors from cluster t_1 and m descriptors from cluster t_2, both n and m being non-zero. The total number of descriptor-descriptor co-occurrence pairs in the domain due to the document are C_2^{n+m}. The number of pairwise inter-cluster co-occurrences due to the document are $C_1^n X C_1^m = nXm$ and the number of pairwise intra-cluster co-occurrences due to the document are C_2^n and C_2^m for the clusters t_1 and t_2 respectively.

According to the new model the contribution of the document towards the similarity weight of a set containing the two clusters t_1 and t_2 is 1. Its contribution towards compactness of the clusters t_1 and t_2 is also 1, provided both n and m are more than 1. If n (or m) is equal to 1, the document has no contribution towards compactness of the cluster t_1 (or t_2).

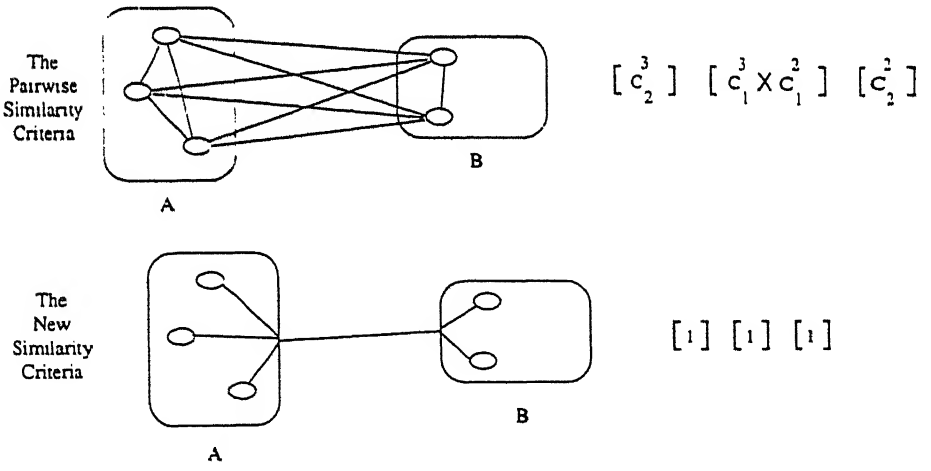


Figure 8: Comparison of the pairwise similarity criteria and the new similarity criteria

The comparison between the old and the new similarity criteria is shown in Fig 8. The figure shows the interaction between two clusters A and B due to a document containing 5 descriptors, 3 from cluster A and 2 from cluster B. On the left hand

side of the old criteria, contribution of the document towards pairwise intra-cluster co-occurrences for A, pairwise inter-cluster co-occurrences and pairwise intra-cluster co-occurrences for cluster B is depicted. On the left hand side of the new criteria, we show the contribution of the document towards the compactness of the cluster A, the similarity weight between the two clusters A and B and the compactness of the cluster B.

Consider a multi-set of documents containing a particular cluster. Let us assume for the simplicity that the cluster is a fundamental cluster. To calculate the compactness of the cluster according to the new model, we can simply count the number of repeatedly occurring elements in the document multi-set. To normalize this count, we divide it by the total number of distinct elements (the total number of documents containing the cluster. We call the resultant number, the multiplicity of the multi-set. We use multiplicity as a clustering criteria for descriptors.

$$multiplicity\ of\ a\ multi-set = \frac{number\ of\ elements\ having\ multiple\ occurrences\ in\ the\ multi-set}{number\ of\ distinct\ elements\ in\ the\ multi-set}$$

The above decisions can be easily extended to non-fundamental clusters - the clusters containing sub-clusters. In this case, we will regard a sub-cluster as a descriptor with a slight variation - All the documents containing one or more number of descriptors of a cluster will be assumed to contain the cluster only once. Thus when this cluster becomes a sub-cluster of another cluster, it will behave just like a single descriptor (which can not be contained in any document multiple times).

Chapter 4

The Algorithm

Having decided about the clustering criteria, the next thing to be decided is the approach for the clustering algorithm.

4.1 Basic Terminology

- **A hierarchical cluster configuration** is a hierarchy of clusters in which each cluster is a collection of descriptors and smaller clusters. Given a set of descriptors there can be a large number of hierarchical cluster configurations.
- **The ideal cluster configuration** is a hierarchical cluster configuration such that the total of multiplicities of all the clusters in the hierarchy is maximum.

We attempt to find an algorithm which gives us the ideal cluster configuration. In the next two sections we present two approaches for building such a configuration using repeated merging of descriptors and repeated splitting of the descriptor-space. In each of the approaches, we point out impracticalities which are, to some extent, applicable to a general class of repeated merging and splitting approaches which may be employed for clustering with our model. Next, we present an incremental algorithm which tries to keep track of a configuration which is slightly inferior to the ideal configuration, as documents get added to the data-space. The incremental approach is actually implemented in our experiments.

4.2 Greedy Approach

1. D is the set of all descriptors in the database.

2. Initially, we have each descriptor in a separate singleton cluster. Thus the set S of all clusters in the cluster-space is

$$S = \{S_\alpha | S_\alpha = \alpha, \alpha \epsilon D\}$$

3. Calculate similarity weights for all possible subsets of S with cardinality two. Now pick the subset S' with maximum similarity weight.

4. Merge the two member clusters of the set S' so as to modify the set of clusters S.

$$S = S - S' \cup \{S'\}$$

5. Repeat the steps 3 and 4 until we get a single cluster containing all the descriptors.

The approach is theoretically correct. But the hitch is, we always get a binary hierarchy because we always check for merging of two clusters at a time. To get an hierarchy which is not necessarily binary, we have to calculate similarity weights for all possible subsets of S (of any cardinality) which is a $\Omega(2^n)$ complex task.

A variation of this approach which uses randomization is as follows -

1. As in the former approach, D is the set of all descriptors. However, no clusters are present initially.

2. Pick a descriptor α randomly from the set of existing descriptors D which are not yet clustered.

3. Create a singleton cluster C containing the descriptor α.

4. We define **a unit transformation on a cluster** as addition or removal of a single descriptor to or from the cluster.

   Find the resultant multiplicity of the cluster C after performing every possible unit transformation on it. Choose the transformation which increases the total multiplicity of the configuration by maximum amount and modify the cluster C accordingly. If there is no unit transformation which increases the total multiplicity, mark the cluster C as non-clusterable, go to step 2.

5. When all clusters are marked as non-clusterable, repeat the above procedure with the set of descriptors replaced by the set of clusters which are formed in this iteration.

6. Stop when a single cluster containing all the descriptors is produced.

Though this method seems nice at a first look, it has a serious problem. As we go on forming clusters, a descriptor (say α) may be clustered incorrectly with descriptor β leading to its unavailability (as a separate descriptor) just at the time when some other descriptor (say γ) which could have formed a better cluster with β, is picked. The problem is due to the fact that we select the best unit transformation in view of the possible unit transformations on the cluster but the transformation may not be the best for the descriptor which is getting added or removed to or from the cluster. In such case, which is highly probable, we may end up in producing a configuration which is nowhere near the ideal configuration in terms of the total multiplicity.

4.3 Repeated Splitting Approach

The problem of clustering can be regarded as **a graph partitioning problem**. We have seen that the document-descriptor domain is really like a graph with documents and descriptors as vertices and occurrences as edges. **A cut-set** is a set of edges in a graph, removal of which leaves the graph disconnected. We define a term **node inclusive cut-set** to denote a set of edges and vertices in a graph, removal of which leaves the graph disconnected. The problem of clustering is the same **as finding**

minimum weight node inclusive cut-sets in the graph. In a document space where every document refers to just one subject, we can find equivalence classes of descriptors using the co-occurrence relation. In such a case, the document-descriptor graph is already disconnected and there is no need to find out minimum weight node inclusive cut-set. In a practical document space, however, there will be some co-occurrence between descriptors of different subject. If we remove some vertices and/or some edges from the graph, we can achieve the configuration in the above scenario. We can then regard the equivalence classes as clusters and the removed descriptors as bridge descriptors.

The scope of the notion of cut-sets is limited to dividing a graph into two partitions. It may be better, in general, to partition the graph into more than two partitions, but there is no clue as to how many partitions will give us the ideal configuration. One can explore a simple binary partitioning which can be described as follows

1. Given a graph of the descriptor-document space, find out a minimum weight node inclusive cut-set

2. Create a cluster containing descriptors represented by the nodes in the node inclusive cut-set. The documents represented by the nodes in the node inclusive cut-set are ignored.

3. Repeat steps 1 and 2 for each of the two partition of the graph as separate document-descriptor spaces. Add the cluster hierarchies created by each of the partitions as sub-clusters to the cluster created in previous step to form a cluster hierarchy of the current document-descriptor space.

The result of this algorithm is a hierarchy of clusters, each of which contains exactly two sub-clusters and zero or more descriptors. To get a hierarchy which is not restrictive in terms of the number of sub-clusters of each cluster, one can explore the possibility of collapsing levels in the resultant binary tree using some criteria (This sort of collapsing of levels in a binary tree is discussed in [MG85], see Chapter 2). However, the level collapsing approach is not justified due to the following reason :

Given that the descriptors α and β are in different partitions in the best binary partition of a given graph, we can not prove that α and β will never be in the same partition in the best n-ary partition of the graph for any n greater than 2.

4.4 Incremental Approach

The task of finding the ideal hierarchy is itself not very clear and hence it is a difficult problem as seen from the previous two sections. We have to compromise, therefore, and attempt to get a hierarchy which is good to some extent. So we define **A Stable Hierarchy** as the one in which total multiplicity can not be increased by moving any single descriptors from its cluster to some other cluster. As one can see, there can be more than one stable hierarchies for a given document-descriptor space.

In the incremental approach, we simulate learning of a stable hierarchy from scratch and as the learning process progresses, we go on feeding documents to the learning algorithm which modifies the existing hierarchical knowledge using the co-occurrences in the new document so as to keep it stable.

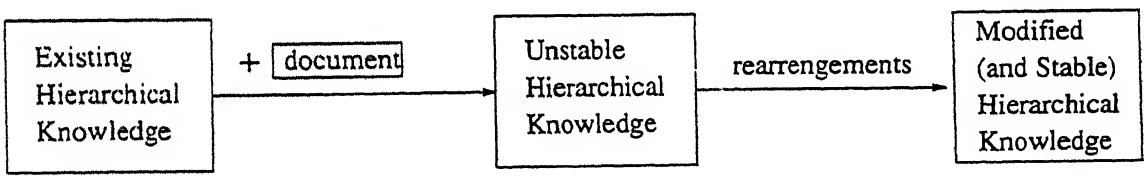


Figure 9: An Overview of Learning Algorithm

Fig 9 summarizes the idea. When a new document is fed to the learning system, the learning algorithm first spots those nodes[1] in the hierarchy which may have become unstable due to the addition of the new document. The rearrangements

[1]Here we interchangeably use the terms 'node' and 'cluster'. The term 'node' is used when we talk in the context of hierarchy and the term 'cluster' is used in the context of descriptor groups and the clustering algorithm. The reader should, however, note that 'node' and 'cluster' are same concepts

giving better configuration are then performed on these nodes of the hierarchy to get a stable hierarchy again.

Firstly, we list out all possible rearrangements in the hierarchy. These are also shown pictorially in Fig 10.

1. Two or more descriptors in a cluster may form a sub-cluster of that cluster.

2. A descriptor in a cluster may move to one of the existing sub-clusters of that cluster. This is because of its occurrence with some descriptor in the sub-cluster.

3. All descriptors in a cluster may move to an existing sub-cluster of that cluster. If it is the only sub-cluster of the cluster, the cluster becomes useless (it does not have any descriptors and has a single sub-cluster and thus it is not clustering anything).

4. a descriptor in a cluster may move to a bigger cluster due to some descriptor in the bigger cluster.

5. Two or more descriptors in different clusters may merge to form a new cluster.

6. Two or more sub-clusters in a cluster may merge to form a single sub-cluster.

We note that all these rearrangements are triggered by co-occurrences in the new document. This is the clue to spot the nodes in the hierarchy which may need rearrangement.

We mark the nodes (clusters) in the hierarchy containing, directly or indirectly, the descriptors present in the new document. The marked clusters themselves form a hierarchical structure as shown in Fig 11. One can traverse this hierarchy top-down or bottom-up in order to check for rearrangements. We choose the top-down approach (specifically Breadth First Traversal). Thus, we can check for major rearrangements (the rearrangements spanning larger part of the hierarchy) first and minor rearrangements (rearrangements spanning smaller parts of the hierarchy) later.

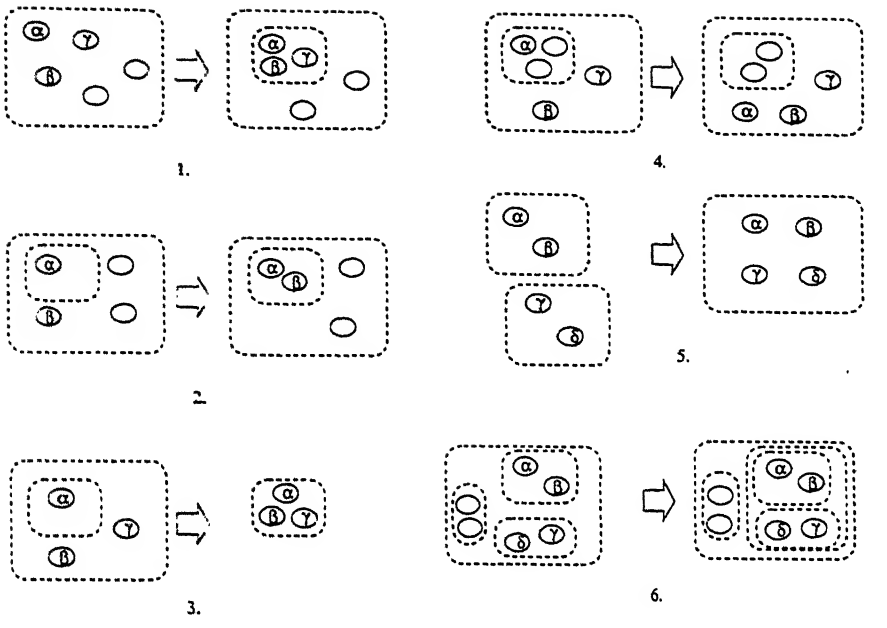


Figure 10: Possible rearrangement for stabilizing cluster hierarchy

The clusters which are of importance to us are branching nodes in this 'marked cluster' hierarchy[2]. This is because, no rearrangement is necessary as long as a set of descriptors refer to the same cluster. It is only when descriptors of different clusters co-occur, that we need some rearrangement. This makes our life even simpler. One just has to traverse the branch nodes in the 'marked cluster' hierarchy.

At an m-b-node, we again need to check only those rearrangements which are restricted to the nodes up to next m-b-nodes. If at all some rearrangement, crossing the boundary of more than one m-b-node needs to be performed, it will be handled by multiple rearrangements, each spanning a range of one m-b-node.

4.5 The Proposed Algorithm

1. Initially, The cluster hierarchy contains a single root node without any descriptors. The root node always represents the whole database.

[2]For convenience, we define a term **m-b-node** for a node in the cluster hierarchy which is itself marked and has multiple marked elements (descriptors and child-nodes).

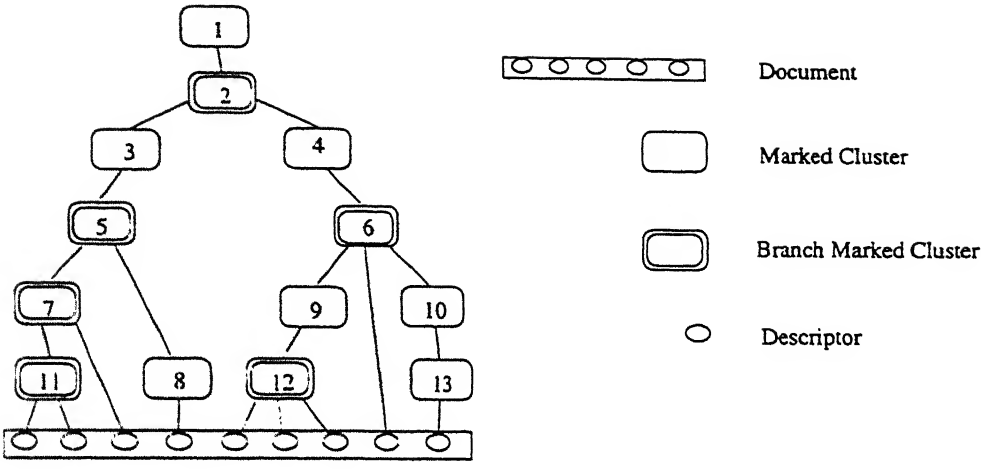


Figure 11: Part of Hierarchical Knowledge affected by a new document

2. Pick new document for processing. Document, in this case refers to a set of descriptors.

3. Mark the clusters in the hierarchy which contain some descriptor of the document.

4. If some descriptors of the document are occurring for the first time, form a new node (cluster) containing all such descriptors. Mark the cluster, and add it as a child node to the topmost m-b-node If there is no m-b-node in the hierarchy, add the new cluster as a child to the bottommost marked node.

5. Traverse m-b-nodes of the cluster hierarchy using Breadth First Traversal. For each m-b-node, perform steps 6 through 8.

6. For each marked child of current m-b-node, note down all nodes up to next m-b-node, in that child branch. We will call this set of nodes as marked-branch of current m-b-node.

7. Check for better configuration (in terms of total multiplicity) of the hierarchy by rearranging the hierarchy in following ways.

   - Moving a descriptor of the current m-b-node to a node in some marked-branch of the m-b-node.

- Moving a node in some marked-branch of the current m-b-node to the m-b-node or a node in some other marked-branch of the m-b-node.

8. Rearrange the hierarchy in best possible configuration among those checked (The one which has the maximum total multiplicity).

9. If due to removal of descriptors or children nodes from a node, the node becomes empty, remove it from the hierarchy. If only one children node and no descriptors remain in some node after rearrangement, attach the only child node to the parent node of the node and remove the node from hierarchy.

10. Repeat rearranging the hierarchy until no further better arrangement is found.

11. Perform the above computation for each new document in the database.

4.6 A Case Study

To understand the algorithm clearly, we present here the scenario of adding an a hypothetical document to the cluster hierarchy.

1. A new document is to be added to the hierarchy. The descriptors of the document are located in the hierarchy.

2. The affected cluster hierarchy is as shown in Fig 11.

3. Start at first m-b-node which is (node 2). marked-branches at (node 2) are {(node 3), (node 5)} and {(node 4), (node 6)}. Check total multiplicity in each of the following configurations

   - (node 3) sub-cluster to (node 4)
   - (node 3) sub-cluster to (node 6)
   - (node 5) sub-cluster to (node 2)
   - (node 5) sub-cluster to (node 4)
   - (node 5) sub-cluster to (node 6)

- (node 4) sub-cluster to (node 3)
- (node 4) sub-cluster to (node 5)
- (node 6) sub-cluster to (node 2)
- (node 6) sub-cluster to (node 3)
- (node 6) sub-cluster to (node 5)

If total multiplicity in any of the configurations is more than total multiplicity in current configuration, change the configuration to the one with maximum total multiplicity and repeat this step for (node 2) again. If no configuration gives better total multiplicity, move to next m-b-node. We assume that no rearrangement is made.

4. Next node in the breadth first traversal of m-b-nodes is (node 5). The marked-branches, here, are {(node 7)} and {(node 8)}. We check following configurations for better total multiplicity

   - (node 7) sub-cluster to (node 8)
   - (node 8) sub-cluster to (node 7)

   Suppose, we get better total multiplicity when (node 8) is attached as a sub-cluster to (node 7). The (node 5) is no longer an m-b-node and we move to next node in BFT which is (node 6).

5. marked-branches at (node 6) are {(node 9), (node 12)} and {(node 10), (node 13)}. Following checks are performed

   - (node 9) sub-cluster to (node 10)
   - (node 9) sub-cluster to (node 13)
   - (node 12) sub-cluster to (node 6)
   - (node 12) sub-cluster to (node 10)
   - (node 12) sub-cluster to (node 13)
   - (node 10) sub-cluster to (node 9)

- (node 10) sub-cluster to (node 12)
- (node 13) sub-cluster to (node 6)
- (node 13) sub-cluster to (node 9)
- (node 13) sub-cluster to (node 12)

In addition configuration changes involving moving of descriptor directly contained in (node 6) to each of (node 9), (node 12), (node 10) and (node 13). Let us assume that there was no better configuration.

6. The next m-b-node in BFT is (node 7). The checks performed are similar to those at (node 6).

7. The next m-b-node in BFT is (node 11) and still next is (node 12). No check is performed in these nodes because these are leaf nodes.

Chapter 5

Implementation and Results

The clustering algorithm discussed in the previous chapter was implemented. It's performance was tested on a sample database. We give here the details of the experimental setup. a general outline of the implementation and the results of the experiment.

5.1 Database Preprocessing

We selected 'Engineering Index - Jan-Mar 97' database for our experiment. A set of bibliographic records which contained the phrases 'computer', 'information technology' or 'software' in their title, keywords or abstract were selected from the database. Only the information about keywords (which are called 'descriptors' and 'identifiers' in Engineering Index terminology) and document location details (title, authors. journal name and issue) was put in ORACLE (Version 7.3.2.3.1) database running on a Pentium machine. The statistics of the database were

Number of documents	:	22488
Number of descriptors	:	118300
Number of descriptor occurrences	:	311872
Average number of descriptors per document	:	13.87
Average number of documents in which a descriptor occurs	:	2.64

It was seen that more than 90% of the keywords occurred just once or twice.

These are not of much use as far as the automatic construction of cluster hierarchy is concerned. Hence we removed all those descriptors which occurred less than 3 times in the database. The statistics of the database after removing these were

Number of documents	:	22487
Number of descriptors	:	5509
Number of descriptor occurrences	:	185044
Average number of descriptors per document	:	8.23
Average number of documents in which a descriptor occurs	:	33.59

The database was stored in terms of three tables :

1. table EIDOC containing the location information for documents such as title, journal name and date of publication.
2. table EIDES containing the descriptors and their cluster identifiers (which were generated as a result of the algorithm) and
3. table EIDD containing the occurrence data (which descriptor occurs in which document).

5.2 Implementation Overview

The implementation was done in *Java* which was primarily chosen because it allows a programmer to build rapid prototypes. The database connectivity offered by *Java* in terms of *JDBC (Java DataBase Connectivity)* was also a consideration.

We give here the outline of the classes implemented in terms of their functionality.

- **Class Clusterer** is the main class which initializes the database, and starts the incremental algorithm.
- **Class Control** reads the documents one by one and adds it to the cluster hierarchy.

- **Class Database** encapsulates details of the bibliographic database. It is used to perform database functions like executing a query, reading previously inferred clustering information from the database. The other important function it performs is providing other classes, an enumeration interface for the documents in the database. The actual database operations are carried out using *JDBC* which we discuss in the next section.

- **Class Descriptor** represents a prototype for a descriptor in the database. It stores information about the cluster it is in and the document it is contained in.

- **Class Document** is used to store document information such as the descriptors it contains. Only one document needs to be stored at a time which is the current document.

- **Class DocumentEnumeration** is the enumerated interface of documents in the database. The characteristic functions include checking whether some more documents are present in the enumeration of documents and fetching the next document in the enumeration.

- **Class Marked** represents a marked cluster in the hierarchy. It is the heart of the algorithm implementation. It contains a routine which Breadth First traverses m-b-nodes in the cluster hierarchy and performs the necessary checks for rearrangements giving a better cluster hierarchy configuration.

- **Class Multiset** is a representation of a multi-set. It is used by the TopicTree Class to store information about which documents contain descriptors of a particular cluster and how many such descriptors each of those documents contain.

- **Class MultisetContainer** is primarily implemented to perform addition and subtraction operation on a multi-set. Its implementation is closely related to the implementation of the Multiset Class. It is also responsible for keeping track of the multiplicity of the multi-set.

- **Class Set** is a representation of a set. It is used by the Descriptor Class to store documents which refer to the particular descriptor. It is also used by the document class to store the descriptors it contains. It is implemented as a linked list.

- **Class TopicTree** represents a cluster of descriptors and sub-clusters (which are again instances of TopicTree Class)

- **Class ShowOut** implements a helper program which outputs the hierarchical information in the database in text form.

- **Class UInterface** implements a primitive user interface to access hierarchical clustering information in the database.

5.3 Database Connectivity - *JDBC*

JDBC is a *Java API (Application Programmer's Interface)* for executing *SQL* statements. It is a package of set of classes and interfaces written in the *Java Programming Language* which makes it possible to write database applications using pure *Java API*. With the *JDBC API*, it is possible to send *SQL* statements to virtually any relational database.

Simply put, *JDBC* supports three kind of functions - establishing a connection with a database, sending *SQL* statements and processing the results.

There are two kinds of *JDBC drivers* available. The first kind of drivers can communicate with the particular database management system being accessed. In the other kind of drivers, commands are sent to a 'middle tier' of services, which then sends *SQL* statements to the actual database.

Our experiments were carried out using *JDBC-ODBC bridge*, which is a part of *Java (Version 1.1)* for *Windows NT*. *JDBC-ODBC bridge* is a piece of code which provides *JDBC* access via *Microsoft's ODBC (Open DataBase Connectivity) drivers*. *ODBC API* is itself a most widely used programming interface for accessing relational databases.

Following are the important methods in the *JDBC driver* which are used in our experiment.

- `Connection DriverManager.getConnection (String url, String username, String password);`
  This method is used to establish a connection (whose id is returned by the method) with a database represented by given url with a given user-id and password. For connection through JDBC-ODBC bridge, the general format of the url is "jdbc:odbc:pc14" where 'pc14' is the name of the database as configured in ODBC driver settings.

  The `DriverManager` class maintains a list of registered `Driver` classes, and when this method is invoked, attempts to locate the appropriate driver that can connect to the database specified by the url.

- `Statement Connection.createStatement ();`
  This method creates a `Statement` object which is used to execute *SQL* query over the given database `Connection` object.

- `ResultSet Statement.executeQuery (String query);`
  and `ResultSet Statement.executeUpdate (String query);`
  These methods are used to execute given *SQL* query through given `Statement` object. The former executes a selection query while the later executes an update, insert, delete or data definition queries.

- `boolean ResultSet.next();`
  Each `ResultSet` object represents results of an *SQL* query, and maintains a cursor which points to current row of result data. This method is used to move that cursor to next row of data. If the operation is successful, the method returns `true`.

- `String ResultSet.getString (int column_number);`
  and `int ResultSet.getInt (int column_number);` These methods are used to read a string and an integer from a given column of a given `ResultSet` object.

5.4 Results

The algorithm was run on the data prepared as mentioned in the first section of this chapter. We give here the results after processing 6000 documents.

To analyze the Hierarchical Clustering process we collected data about the time required and various operations performed. In the first section we will investigate the nature of this accounting data and in next section we will present the actual result of the clustering process : The cluster hierarchy.

5.4.1 Process Trends

Fig 12 shows the number of descriptors added in the knowledge base while processing the documents. The nature of the curve needs a little explanation. As more and more documents get processed, more and more descriptors from new documents are found in the hierarchy and processing required for adding new descriptors in the hierarchy decreases.

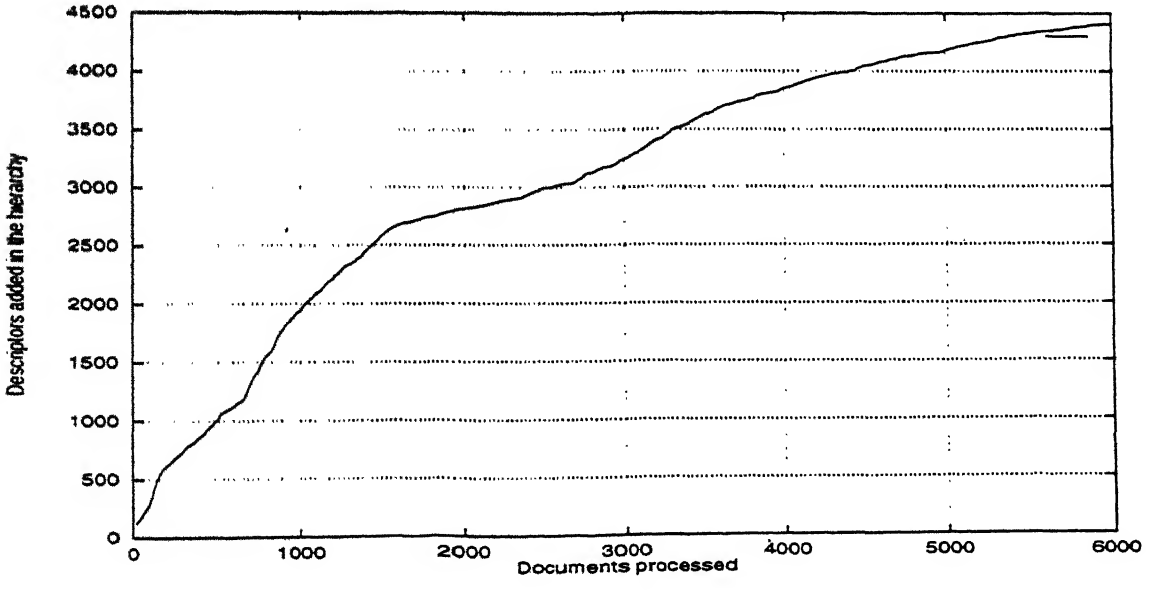


Figure 12: Trend - Number of descriptors in the hierarchy

Next, we investigate the number of m-b-nodes at each point of time (see Fig 13.

This value is 1. directly proportional to the number of descriptors for a document and 2. inversely proportional to how well the descriptors assigned to a document agree with the the categorization information in the hierarchy. This dependence is due to the fact that m-b-nodes are the nodes at which the document descriptors get separated into two or more groups (see Fig 11).

The first quantity above is a random variable. The second quantity should increase as we are using descriptor information while forming the hierarchy. However, as there is no centralized system which assigns descriptors to all documents, the increase may not be significant.

The trend for number of m-b-node shows a random behavior which is justifiable due to the randomness of number of descriptors. It also shows a slight increase which is contradictory to above explanation. However, we can not conclude anything because the trend of new descriptors added to the database is not saturated yet and the hierarchy is not yet fully formed.

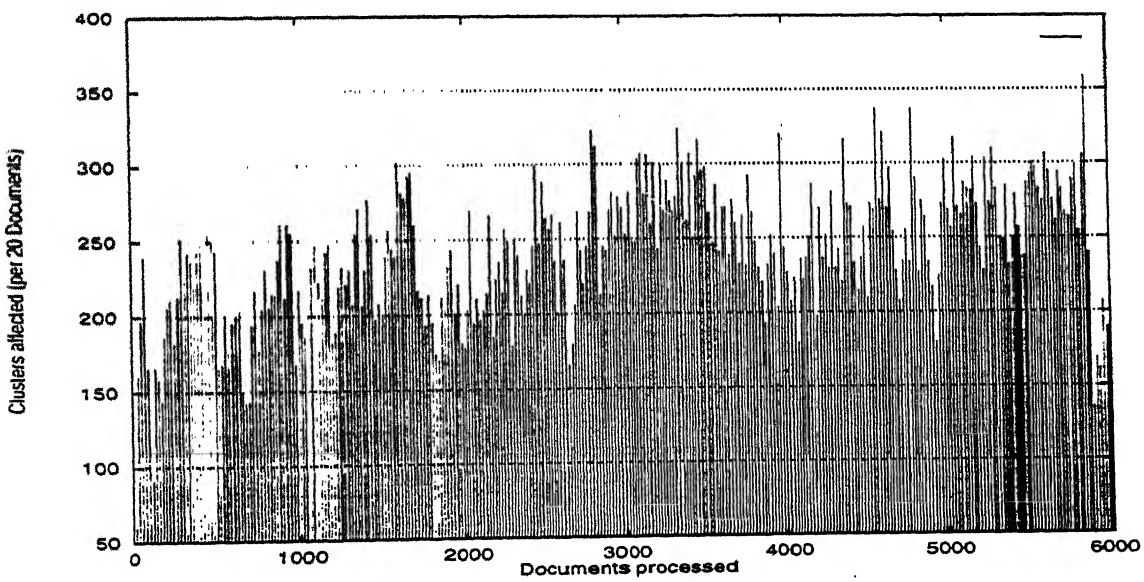


Figure 13: Trend - Number of m-b-nodes

Fig 14 shows the number of checks performed for rearrangement of affected clusters in the cluster hierarchy to get a better configuration in terms of total multiplicity

of the hierarchy. This number is dependent to some extent on the number of m-b-nodes and the number of descriptors in a document (These are the only components of the hierarchy which are checked against rearrangement for a configuration with higher multiplicity). The trend is increasing in the initial stages of building the hierarchy. The rate of increase is decreasing in the later part. It is expected to remain constant once number of descriptors saturate.

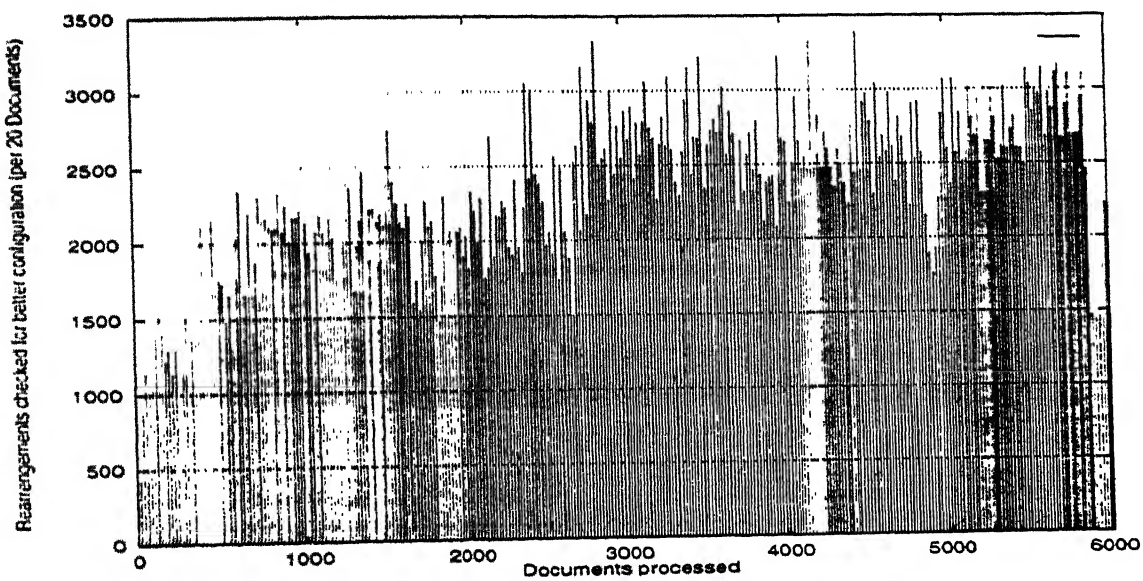


Figure 14: Trend - Number of checks for rearrangement in the hierarchy

Fig 15 shows the actual number of rearrangements done in the hierarchy. In this figure, one can notice a cyclic behavior. The expected characteristic is a decreasing trend because the number of rearrangements truly reflects the stability of the hierarchy. The cyclic nature may be attributed to periodic increase in rearrangements of clusters which are formed earlier in the light of the information available at that time but have become unstable now.

The overall number of rearrangements is plotted in Fig 16. The figure shows a linear relationship with the number of documents. The number of rearrangements done for clustering is, thus, more or less constant.

A major chunk of the time required to process documents is spent in checking

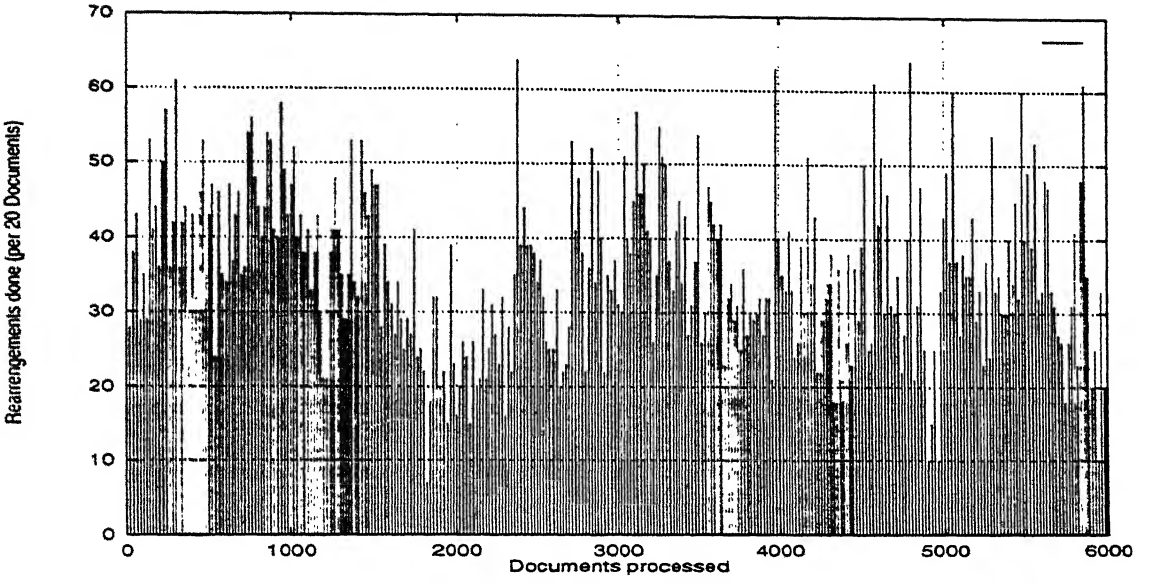


Figure 15: Trend - Number of rearrangements done in the hierarchy

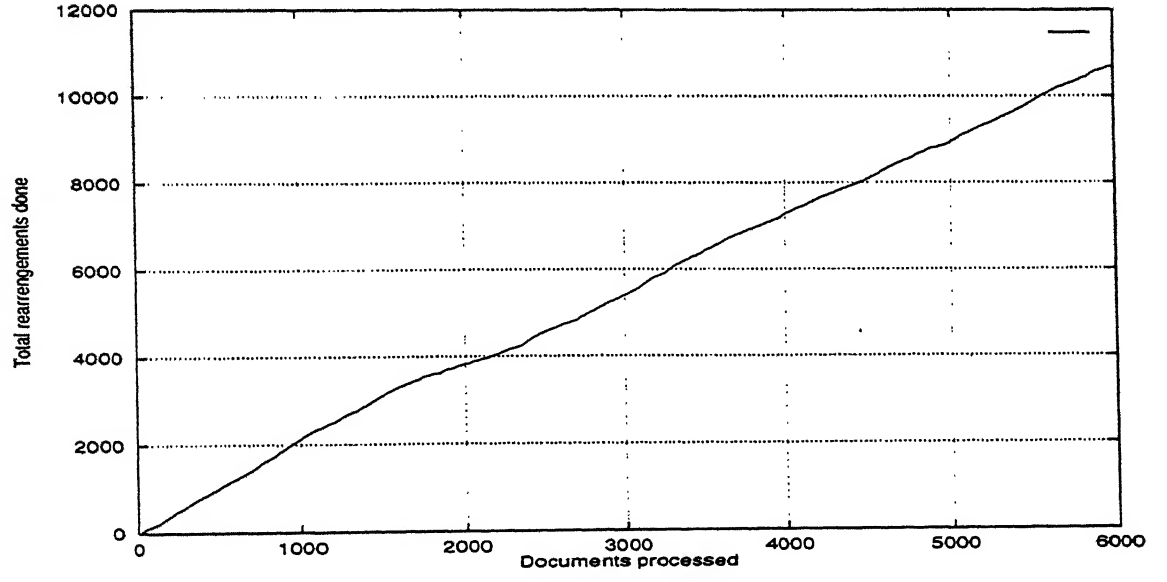


Figure 16: Trend - Overall (total) number of rearrangements done in the hierarchy

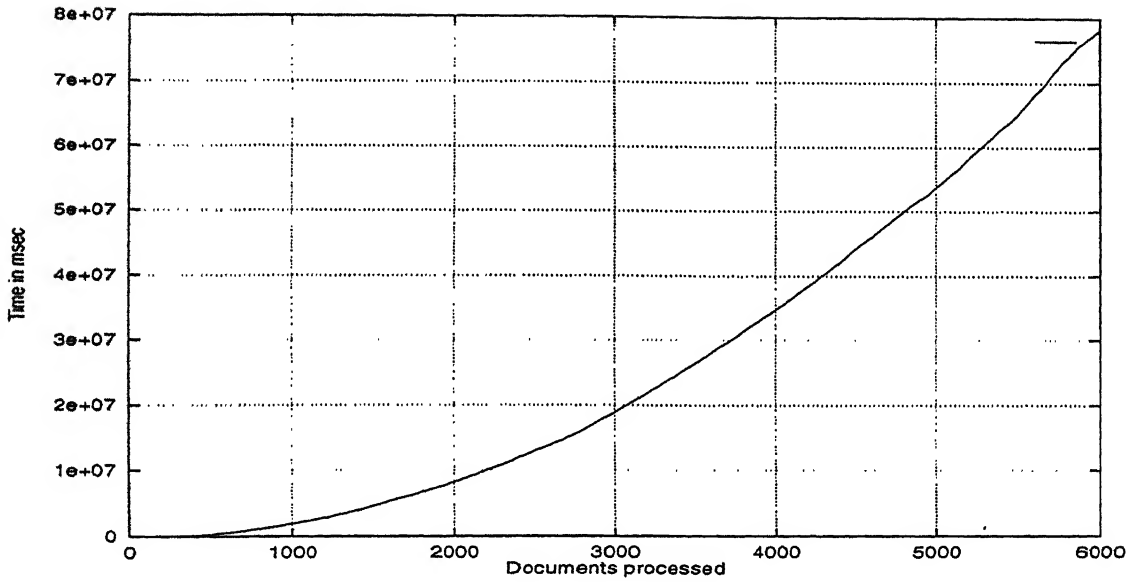


Figure 17: Trend - Processing Time

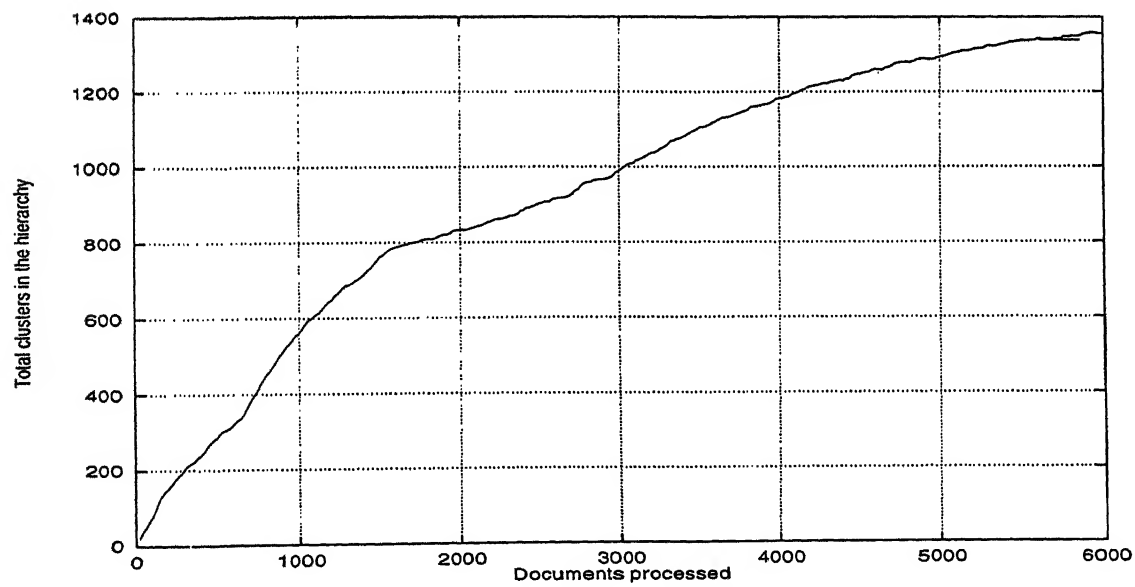


Figure 18: Trend - Number of clusters in the Hierarchy

various configurations for better total multiplicity. Checking for a configuration involves calculating the union of two multi-sets of documents. To get a rough estimate of the time complexity of the algorithm, we have to consider the number of times the union operation is performed and the time required for each union calculation operation. We have already seen that the number of checks performed are constant with respect to time. Hence it is the time required for each union operation which governs the time complexity of the algorithm. The union operation is implemented as merging of two linked-lists which has a time complexity of $O(m+n)$ where m and n are the lengths of the two linked lists.

In summary, the algorithm spends $O(m+n)$ time for each document, where m and n both depend on the number of processed documents. The algorithm is thus expected to have $O(n^2)$ behavior.

The actual time required for processing documents in the experimentation is shown in Fig 17. It shows $O(n^2)$ complexity which is expected. Research in the past [Cro90] has shown that clustering is an $O(n^2)$ process. Analysis of our algorithm and the experimental results confirm the fact.

Finally, the number of clusters formed in the hierarchy is depicted in Fig 18. The curve is similar to the one for number of descriptors. It is seen that the number of clusters is proportional to the number of descriptors in the hierarchy, which is intuitive.

5.4.2 Cluster Hierarchy

To understand the overall structure of the resultant cluster hierarchy, we collected information about distribution of descriptors and the clusters formed in the hierarchy. It is summarized in Table 1 and 2. Following facts about the hierarchy were revealed by this information.

1. The hierarchy has 18 levels including the root cluster. The average number of descriptors in a cluster is 3.163. The average number of sub-clusters per cluster is 2.344 and the average multiplicity of a cluster is 0.245. Thus the clusters do exhibit a hierarchical structure.

Fundamental clusters					
Level	# of clusters	# of desc.s	(per cluster)	Multiplicity Total	(Ave.)
0	0	0	(-)	0.000	(-)
1	24	46	(1.917)	15.667	(0.653)
2	22	51	(2.318)	9.937	(0.452)
3	20	46	(2.300)	8.726	(0.436)
4	23	56	(2.435)	10.222	(0.444)
5	32	77	(2.406)	9.386	(0.293)
6	58	142	(2.448)	13.597	(0.234)
7	70	185	(2.643)	22.810	(0.326)
8	96	289	(3.010)	21.908	(0.228)
9	127	317	(2.496)	42.216	(0.332)
10	74	200	(2.703)	20.712	(0.280)
11	92	284	(3.087)	25.533	(0.278)
12	76	218	(2.868)	22.984	(0.302)
13	43	138	(3.209)	13.969	(0.325)
14	15	50	(3.333)	3.489	(0.233)
15	4	10	(2.500)	0.737	(0.184)
16	0	0	(-)	0.000	(-)
17	1	2	(2.000)	0.333	(0.333)

Non-fundamental clusters					
Level	# of clusters	# of desc.s	(per cluster)	Multiplicity Total	(Ave.)
0	1	107	(107.000)	0.648	(0.648)
1	5	70	(14.000)	1.540	(0.308)
2	7	36	(5.143)	2.248	(0.321)
3	13	84	(6.462)	1.727	(0.133)
4	29	81	(2.793)	3.523	(0.121)
5	39	116	(2.974)	4.820	(0.124)
6	50	193	(3.860)	7.046	(0.141)
7	74	261	(3.527)	9.376	(0.127)
8	81	327	(4.037)	11.100	(0.137)
9	69	212	(3.072)	11.096	(0.161)
10	78	278	(3.564)	11.378	(0.146)
11	74	230	(3.108)	11.723	(0.158)
12	39	119	(3.051)	10.194	(0.261)
13	12	38	(3.167)	2.298	(0.192)
14	5	15	(3.000)	0.927	(0.185)
15	1	2	(2.000)	0.200	(0.200)
16	1	6	(6.000)	0.167	(0.167)

Table 1: Distribution of descriptors in the cluster hierarchy

Total clusters					
Level	# of clusters	# of desc.s	(per cluster)	Multiplicity Total	(Ave.)
0	1	107	(107.000)	0.648	(0.648)
1	29	116	(4.000)	17.206	(0.593)
2	29	87	(3.000)	12.185	(0.420)
3	33	130	(3.939)	10.453	(0.317)
4	52	137	(2.635)	13.745	(0.264)
5	71	193	(2.718)	14.205	(0.200)
6	108	335	(3.102)	20.644	(0.191)
7	144	446	(3.097)	32.186	(0.224)
8	177	616	(3.480)	33.008	(0.186)
9	196	529	(2.699)	53.312	(0.272)
10	152	478	(3.145)	32.089	(0.211)
11	166	514	(3.096)	37.256	(0.224)
12	115	337	(2.930)	33.178	(0.289)
13	55	176	(3.200)	16.268	(0.296)
14	20	65	(3.250)	4.416	(0.221)
15	5	12	(2.400)	0.937	(0.187)
16	1	6	(6.000)	0.167	(0.167)
17	1	2	(2.000)	0.333	(0.333)

Table 2: Distribution of descriptors in the cluster hierarchy - *Cont'd.*

Consider a database in which there is no variation in number of co-occurrences between descriptor pairs, triplets, (and so on). Due to this, the total multiplicity of an average hierarchical cluster configuration will be very close to that of the ideal cluster configuration and the probability of getting the ideal cluster configuration also becomes less. The depth of the hierarchy will also reduce in this case. A deep hierarchy is thus an indication of the variation in number of occurrences and co-occurrences among the descriptors.

2. The average number of descriptors in a cluster is independent of the level of the cluster.

   This is basically due to variety of descriptors in terms of the generality. For example, descriptors as broad as 'computer networks' or 'database systems' and as narrow as 'Bibliographic retrieval system' or 'Automatic test pattern

generation' are present in the database.

3. The average multiplicity of fundamental clusters is considerably higher than the average multiplicity of non-fundamental clusters. However, the multiplicity of a cluster is independent of the level of the cluster in the hierarchy.

   This is because no explicit condition favoring higher multiplicity in lower levels is incorporated in the model. Ideally, we would like to have increase in the average multiplicity as we go down the hierarchy. This would enable the specific clusters (clusters lower in the hierarchy) to be more compact than generic clusters (which are higher in the hierarchy). Further investigation is needed in the direction of refining the model to incorporate such a behavior.

It is impossible to give the entire cluster hierarchy here. In Fig 19 we give some representative samples of the hierarchy. Notable facts about the cluster hierarchy are as follows :

1. The leaf clusters of the hierarchy show descriptors which are semantically closely related.

2. In some clusters, it is seen that descriptors from two semantic topics are co-existing. Further investigation shows that some other descriptors from those two topics (semantic) are forming sub-clusters to the cluster. For example see figure 19, third sample. descriptors from two topics viz. computer networks and computer systems are coexisting.

   This problem is due to insufficient information about the descriptors. Unless documents of each of these two topics without any descriptor of the other topic do not enter the hierarchy, we will not get the clean separation in terms of clusters.

3. When smaller parts of hierarchy are examined, they contain descriptors which are semantically close but when the hierarchy is considered as a whole, it does not exhibit a structure similar to the semantic hierarchy.

Topic	[..29.6.5.5.2.1.2]	Open architecture
Topic	[..29.6.5.5.2.1.2.1]	Software reuse
Topic	[..29.6.5.5.2.1.2.1.1]	Context sensitive languages
Topic	[..29.6.5.5.2.1.2.1.1.1]	Reusable software components, Systems engineering
Topic	[..29.6.5.5.2.1.2.1.1.2]	Data description, Requirements engineering
Topic	[..29.6.5.6.1]	Curve fitting, Self adjusting control systems
Topic	[..29.6.5.6.1.1]	Dies, Forging, Forgings
Topic	[..29.6.5.6.1.2]	Automobile engines, Fuel injection, Rapid prototyping
Topic	[..29.6.5.6.1.3]	Automobile electronic equipment, Intake systems, Microelectromechanical devices, Transducers
Topic	[..29.6.5.6.1.4]	Computer aided engineering, Gear manufacture
Topic	[..29.6.5.6.1.5]	Robot control, Torque
Topic	[..29.6.5.6.1.5.1]	Reluctance motors, Rotors (windings), Vibration measurement
Topic	[..29.7.6.1.3.5]	Computer networks, Data distribution, Intelligent networks, Interconnection networks, Multidatabase systems
Topic	[..29.7.6.1.3.5.1]	PROM, Service control point
Topic	[..29.7.6.1.3.5.2]	Computer operating systems, Computers, Distributed database systems
Topic	[..29.7.6.1.3.5.2.1]	Direct sequence spread spectrum, Modems
Topic	[..29.7.6.1.3.5.2.2]	Automotive industry, Cost reduction, Improvement
Topic	[..29.7.6.1.3.5.2.3]	Client server, Computer operating procedures, DOS
Topic	[..29.7.6.1.3.5.2.4]	Intellectual property, Internet, Smart cards
Topic	[..29.7.6.1.3.5.2.4.1]	Broadcasting, Telecommunication
Topic	[..29.7.6.1.3.5.2.4.2]	Magnetic resonance spectroscopy, Purchasing, Wide area networks, World wide web (WWW)
Topic	[..29.7.6.1.3.5.2.4.2.1]	Copyrights, Image sequences, Internet service providers (ISPs)
Topic	[..29.7.6.1.3.5.2.4.2.2]	Error correcting codes, Security of data
Topic	[..29.7.6.1.3.5.2.4.2.2.1]	Automatic teller machines, Computer crime
Topic	[..29.7.6.1.4.4]	Acceleration, Gravitation
Topic	[..29.7.6.1.4.4.1]	Image warping, Parameter extraction
Topic	[..29.7.6.1.4.4.2]	Electromagnetic fields, Slot antennas
Topic	[..29.7.6.1.4.4.3]	Portable equipment, Video signal processing
Topic	[..29.7.6.1.4.4.3.1]	Computer terminals, Interactive devices
Topic	[..29.7.6.1.4.4.3.2]	Special effects, Television broadcasting
Topic	[..29.7.6.1.4.4.3.2.1]	Computer generated holography, Holographic optical elements, Lithography, Masks, Optical recording
Topic	[..29.7.6.1.4.4.3.3]	Character recognition, Display devices
Topic	[..29.7.6.1.4.4.3.3.1]	Optical filters, Optoelectronic devices, Television systems
Topic	[..29.7.6.1.4.4.3.4]	Content based image retrieval, Multimedia systems
Topic	[..29.7.6.1.4.4.4]	Health care, Medical applications, Noninvasive medical procedures, Surgery
Topic	[..29.7.6.1.4.4.4.1]	Dentistry, Diagnostic radiography
Topic	[..29.7.6.1.4.4.5]	Infrared detectors, Sensor data fusion
Topic	[..29.7.6.1.4.4.5.1]	Image sensors, Stereo vision, Video cameras
Topic	[..29.7.6.1.4.4.6]	Free surface, Velocity

Figure 19: Resultant cluster hierarchy

Chapter 6

Conclusion and Future Work

6.1 Summary

Information retrieval has always been a challenge for an information seeker. The major problems faced by the user are the retrieval of irrelevant results along with relevant results and poor percentage of retrieval of relevant information as compared to total relevant information in the database. Analyzing both these problems was the primary aim behind this thesis. Towards this, we first investigated the possibility of providing a better user interface to help the user specify his interest in some kind of a 'user profile'. Also related to this was the issue of getting a good navigation tool/guide, which could be integrated with such a user interface to dis-ambiguate user specifications and help the user understand the organization of the domain knowledge more clearly. This thesis has tried to build such a navigation tool.

We propose an algorithm to infer clusters of document descriptors in the database, and thereby form a hierarchy of clusters which has a structure very similar to a conventional thesaurus.

6.2 Conclusive Remarks

The proposed model for clustering a descriptor space to get a hierarchical structure of clusters is totally based on the assumption that the descriptors of the documents are good enough to represent them semantically. However, this is not always the

case. One of the directions for future work could be development of an intelligent descriptor generator for documents.

The attempt to build a cluster hierarchy was successful to some extent. However, there are some limitations (mentioned below) and a good amount of effort is still necessary in order to judge the usefulness of the model developed.

The current clustering algorithm forms good clusters at the leaf nodes of the hierarchy. The compactness of higher level clusters, however, needs some attention. The problem with these clusters is the improper grouping of clusters to form larger clusters. There are some cases in which a set of descriptors which should have been clustered into two or three separate clusters are wrongly grouped to form a single cluster. So the clustering criteria needs greater refinement.

Developing a good user interface to access the cluster information is one of the possible extensions to this work. Such an interface should be designed around the basic principal of our model - *A document typically refers to more than one subject.* An information delivery system discussed in the beginning of Chapter 3 could be implemented using the cluster hierarchy generated here. A user's profile can be maintained in terms of known clusters in the hierarchy. The main problem which needs to be addressed, if such a system is to be implemented, is developing a model for tracking the clusters as they get rearranged due to addition of more and more documents to the database.

Bibliography

[AF81] R. Attar and A.S. Fraenkel. Experiments in local metrical feedback in full-text retrieval systems. *Information Processing and Management*, 17:115–126, 1981.

[Ass] Athenia Associates. Search engines. *http://webreference.com/content/search.*

[Bat79] M. Bates. Information search tactics. *Journal of American Society of Information Science*, pages 205–214, 1979.

[BM85] D. C. Blair and M. E. Maron. An evaluation of retrieval effectiveness for a full-text document retrieval system. *Communications of the ACM*, 28(3):289–299, 1985.

[Bor86] C. L. Borgman. Why are online catalogs hard to use? *Journal of American Society of Information Science*, 37(6):387–400, 1986.

[CFMR89] Robert S. Tamaru Charles F. McMath and Roy Rada. A graphical thesaurus-based information retrieval system. *International Journal of Man-Machine Studies*, 31:121–147, 1989.

[CL92] Hsinchun Chen and Kevin J. Lynch. Automatic construction of network of concepts characterizing document database. *IEEE trans. on Systems, Man and Cybernetics*, 22(5):885–902, September/October 1992.

[Cro90] C. J. Crouch. An approach to the automatic construction of global thesauri. *Information Processing and Management*, 26(5):629–640, 1990.

[CT87] W. B. Croft and R. H. Thompson. I^3R: A new approach to the design of document retrieval systems. *Journal of American Society of Information Science*, 38(6):389–404, 1987.

[GS91] Susan Gauch and John B. Smith. Search improvement via automatic query reformulation. *ACM trans. on Information Systems*, 9(3):249–280, July 1991.

[GSY75a] A. Wong G. Salton and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18:613–620, 1975.

[GSY75b] C. S. Yang G. Salton and C. T. Yu. A theory of term importance in automatic text analysis. *Journal of American Society of Information Science*, 26(1):33–44, 1975.

[HCN93] Koushik Basu Hsinchun Chen, Kevin J. Lynch and Dorbin Ng. Generating, integrating and activating thessauri for concept-based document retrieval. *IEEE Expert*, pages 25–34, April 1993.

[KDD] KDD. Kd mine: Data mining and knowledge discovery resources home. *http://info.gte.com/ kdd/index_kdm.html.*

[MC87] I. Monarch and J. G. Carbonell. Coalsort: A knowledge-based interface. *IEEE Expert*, 2(1):39–53, spring 1987.

[MG85] Tong R. M. and Shapiro D. G. Experimental investigations of uncertainty in a rule-based system for information retrieval. *International Journal of Man-Machine Studies*, 22:265–282, 1985.

[OS82] B. K. Oldroyd and J. J. Schroder. Study of strategies used in online searching: 2. positional logic - an example of the importance of selecting the right boolean operator. *Online Review*, 6(2):127–133, 1982.

[PJSC89] Deb Galdes Philip J. Smith, Steven J. Shute and Mark H. Chignell. Knowledge-based search tactics for an intelligent intermediary system. *ACM trans. on Information Systems*, 7(3):246–270, July 1989.

[PJSK84] Mark H. Chignell Philip J. Smith and D. A. Krawczak. Developement of a knowledge-based bibliographic information retrieval system. *Proc. of IEEE conference on Systems, Man and Cybernetics*, pages 222–225. Oct 1984.

[Ris84] E. Rissland. Ingredients of intelligent user interfaces. *International Journal of Man-Machine Studies*, 21:377–388, 1984.

[Sho85] P. Shoval. Principles, procedures and rules in an expert system for information retrieval. *Information Processing and Management*, 21(6):475–487. 1985.